N1S01-CORE-5.0

单串口以太网络服务器 5.0 版核心模块

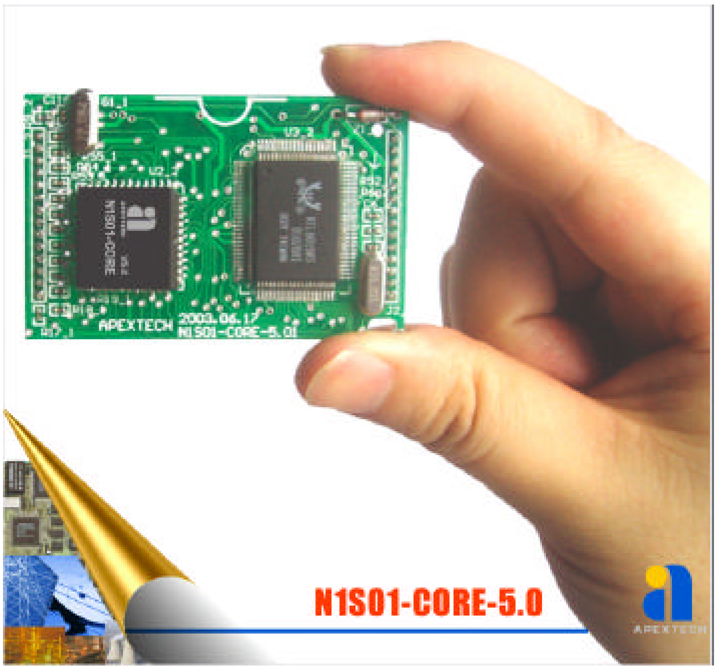
概述

N1S01-CORE-5.0是一个设计应用于单片机系统网络接口的功能完善的TCP/IP协议栈模块，主要完成了串口信息和网络信息之间的双向透明交互，同时提供了两路用于远程检测与控制目的输入/出信号。

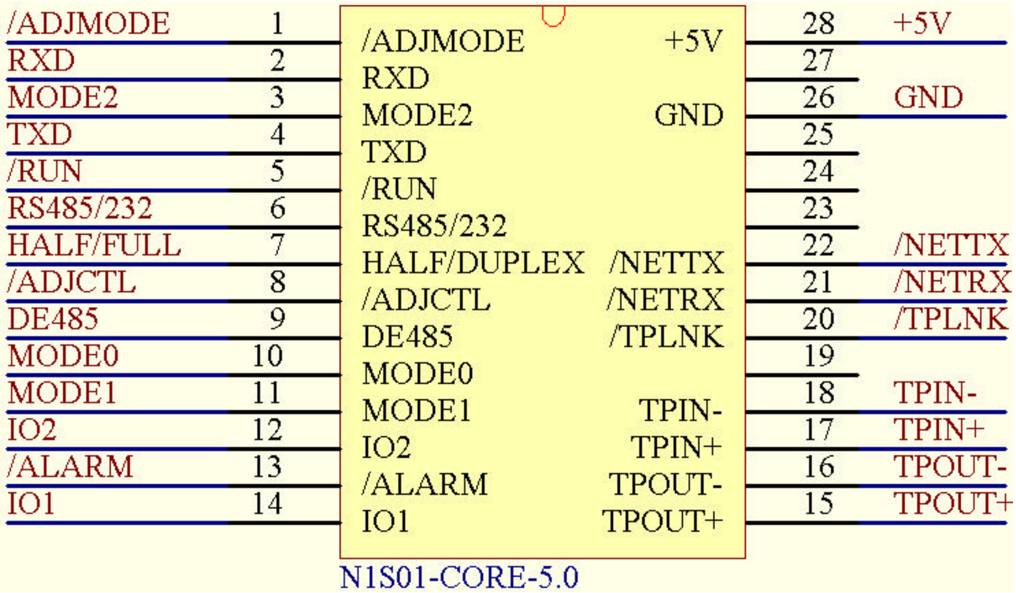
与其他型号的产品相比，具有性价比良好、网络链接可靠、应用接口灵活等优点。

主要特点

- 1 个串口，支持 300bps～115200bps 的波特率；7、8 位数据位；1、2 位停止位；奇校验、偶校验、无校验；
- 1 个 10BaseT 网口，与 IEEE802.3 兼容；支持 ICMP、ARP、PING、TCP/IP 协议或服务（可以根据用户要求开放 UDP、Telnet）；
- 网络链接可靠恢复技术的应用，能在几秒钟内恢复任何原因丢失的网络链接；
- 满足 RS232/485/422 灵活接口要求，具备全/半双工自动控制能力；
- 无须编写网络通信代码的透明传输方式，可主动或被动完成 TCP/IP 链接；
- 提供两路通用输入/出，可以用作远程状态检测和控制（如继电器）；
- 提供网络数据收发状态和外部网络连接状态信号指示；
- 为模块和网络异常状态提供告警信号输出；
- 网络信号输出采用焊接，适应于二次系统设计时的不同结构安排，并提高可靠性；
- 低功耗 5V@50mA 的芯片式设计，可在线修改参数，尤其适用于单片机系统；
- 工作环境：－25℃～＋75℃、20%～90%RH、不结露；
- 保存环境：－40℃～＋85℃、20%～95%RH，不结露；
- 封装形式：2 排 14 针、针间距 2.0mm；
- 模块尺寸：4cm×6cm。



引脚定义



引脚	信号	方向	说明
1	/ADJMODE	输出	0- 模块处于参数设置/调试状态； 1- 模块处于正常运行状态；
2	RXD	输入	模块串口 TTL 接收信号；
4	TXD	输出	模块串口 TTL 发送信号；
5	/RUN	输出	0、1 交替变化，作为模块 CPU 运行状态指示； /ADJMODE 为 0 时，/RUN 变化频率为 1Hz； /ADJMODE 为 1 时，/RUN 变化频率为 2Hz；
6	RS485/232	输出	0- 模块采用 RS232 外部接口，全双工方式； 1- 模块采用 RS485 外部接口，全半双工由 HALF/FULL 选择； 本信号由模块根据设置参数自动输出；
7	HALF/FULL	输出	0- RS485 工作在全双工模式； 1- RS485 工作在半双工模式； RS485/232 为 0 时，本信号可以忽略；本信号由模块根据设置参数自动输出；
9	DE485	输出	0- RS485 发送禁止； 1- RS485 发送使能； HALF/FULL 或者 RS485/232 为 0 时，本信号可以忽略；本信号由模块根据设置参数自动输出；

8	/ADJCTL	输入	0- 持续该状态 100ms 将使模块进入参数设置/调试状态； 1- 持续该状态 500ms 将使模块返回正常工作状态； 本信号是否起作用受设置参数控制，参见后面的设置参数定义；
10、11、3	MODE0、MODE1、MODE2	输入	模块测试用途；应悬空；
14、12	IO1、IO2	准双向	与 8051 的 P1 口原理相同；输入或输出，即远程可读取引脚上的电平状态，以便完成状态量的检测；也可在此引脚输出指定的电平，以便执行控制等功能；
13	/ALARM	输出	0- 模块自检异常，可进入调试状态取得异常信息报文，查找原因； 1- 模块自检正常；
15、16	TPOUT+、TPOUT-	输出	网络差分输出信号；
17、18	TPIN+、TPIN-	输入	网络差分输入信号；
20	/TPLNK	输出	0- 外部网络线连接正常； 1- 外部网络线异常；
21	/NETRX	输出	0- 网络接收数据；每接收一帧该信号大约持续有效 100ms； 1- 无效；
22	/NETTX	输出	0- 网络发送数据；每发送一帧该信号大约持续有效 100ms； 1- 无效；
26	GND		电源地
28	+5V	输入	电源输入，+5V±5%、50mA
19、23、24、25、27	N.C		空脚；应悬空。

串口与 IP 的关系

1、串口与 IP 端口映射

N1S 全系列的串口设备以太网络服务器采用 IP 端口映射技术来描述串口与 IP 之间的关系，即串口设备以太网络服务器的所有串口共享一个 IP 地址，各串口分别对应于同一 IP 下的一个端口。从 ApexTech Setting 设置管理中可知，假设串口设备以太网络服务器的 IP 地址为 IP1，基础端口号为 PORT，则存在如下对应关系：

COM1-IP1、PORT，COM2-IP1、PORT+1，……，COMn-IP1、PORT+n

假设网络内的 IP 地址为 IP2 的计算机/终端需要访问串口设备以太网络服务器的串口 1，则实际数据链路关系为：

IP2+某端口 ←→ IP1+PORT←→ 串口 1

2、WinSock 链接的保持

从串口与 IP 端口的映射关系可知，在网络内访问串口设备以太网络服务器的串口，实际工作为：

串口设备以太网络服务器的串口	计算机/终端
● 根据串口所接入 DTE 的通信参数进行设置； ● 确认串口、网络线等链接正确，然后上电，工作于指定的工作模式，并等待（或尝试）TCP/IP 链接（请求）。	● 确定串口设备以太网络服务器的 IP 地址 IP1 和所需访问的串口对应（映射）的端口号 PORT； ● 确定本机的 IP 地址 IP2 和用于建立链接的端口号 PORT2； ● 设置 Winsock 的相关参数，尝试（或等待）建立链接。

当串口设备以太网络服务器工作在“工作模式一”时，其上电后即运行在服务器模式，即创建了套接字，并随时侦听链接请求；当收到客户机链接请求时，接受链接并保持链路畅通，直至客户机中断（关闭）链接或网络路由中断并超时；当链路不管任何原因关闭（中断）后，串口设备以太网络服务器将重新回到侦听链接请求的工作状态。

当串口设备以太网络服务器工作在“工作模式二”时，其上电后即开始尝试建立各串口指定的链路，即工作在客户机模式。根据各串口指定的 IP 地址和端口号，串口设备以太网络服务器尽力保持链路的畅通（包括链路中断后的重试）。

不管串口设备以太网络服务器工作在服务器，还是客户机模式，一旦链路畅通，任何来自网络的与串口映射对应的端口的信息将立即送到对应的串口，同时串口收到的信息也将被立即送到对应的链路中；从而从外部看来，串口设备以太网络服务器的串口和网络上的计算机之间形成了一条与具体 DTE 装置通信参数无关的透明、全双工的数据链路。

重要提示

·对于需要长期保持链路畅通的应用，若串口设备以太网络服务器工作在服务器模式，则客户机要随时检查链路的可用性，并负责重新与串口设备以太网络服务器建立链路；若串口设备以太网络服务器工作在客户机模式，则由串口设备以太网络服务器负责维护数据链路；

· 各串口所建立的链路相对独立，不会因为一个链路的关闭导致其它链路中断；

· 对于串口设备以太网络服务器所接 DTE 设备需要多方共享的情况，建议采用“工作模式一”，即串口设备以太网络服务器工作在服务器模式，由访问方负责创建并维护数据链路，当访问操作完成时应及时关闭链接，以便让别的客户机有机会访问串口设备以太网络服务器的同一个串口，即同一个 DTE 设备，因为全双工方式下多个客户机同时通信是有冲突的，因此只能分时工作（访问）；

· 对于“只发不收”的情况，由于不存在数据混乱的可能性，因此可以同时建立多个数据链路，以满足“一发多收”的需求；当然，“一发多收”还可以通过建立“IP 多播”以更优的方式来完成；

· 对于“只收不发”的情况，由于通信协议一般是按帧（多字节）进行的，因此同时接收多方数据极易导致信息帧错误，因此需要建立可靠的通信（任务）“开始”与“停止”的控制，建议不要采用这种通信方案，若必须采用时请与我们联系，以便为您定制可靠的服务程序；

· 虽然 TCP/IP 是相对严格的有可靠性保证的通信协议组，但由于网络的影响、操作系统和高层应用软件的可靠性、完善程度等原因，仍存在数据包丢失的可能性，因此高层应用软件的具体应用协议应对数据的完整性给出检查和纠错方案，简单的纠错方案就是“数据确认”和“数据重发”；

· 由于串口速率的限制，通过网络高速发往串口设备以太网络服务器对应串口的信息

量要加以限制，否则可能因为缓冲溢出导致数据包丢失。

3、数据的传输

由于数据在链路上的传输是"透明的"，因此对于串口设备以太网络服务器所接入的 DTE 设备，无须修改任何程序，而对于原来采用串口与 DTE 设备通信的应用程序而言，只需要将串口通信的端口访问部分改成 Winsock 通信即可，而原来设计的数据收发缓冲和报文解释均无须修改，这样可以满足同一高层管理程序即可采用串口通信、又可满足 Ethernet/Internet/Intranet/ATM 等的网络通信通信需求，具体如下：

	串口通信	网络通信
通信参数	通过各种编程方式直接或间接完成波特率等串口通信参数的设置；	通过设置 IP 地址等参数直接或间接调用 Winsock 完成网络数据链路的建立；
数据传输	直接或间接读写串口；	标准的 Winsock 接口 API 函数或 Winsock 控件属性（方法）的访问；
收发缓冲	各串口均拥有专用的一个接收缓冲和一个发送缓冲；	各链路均拥有专用的一个接收缓冲和一个发送缓冲；
	因此可以相互替代，而满足串口或网络通信需求；	
数据收发	通过查询或事件驱动（回调函数）等方法完成；也可采用 MSComm 控件等方法；	通过查询或事件驱动（回调函数）等方法完成；也可采用 Winsock 控件等方法；
报文处理	可公用一个报文组织和解析流程；	
链路关闭	可以简单终止接收或发送数据即可；	标准的 Winsock 接口 API 函数或 Winsock 控件属性（方法）的访问；

确定您的应用需求

1、数据流向的确定

使用串口设备以太网络服务器时，在设置工作模式之前应先确定具体应用中的数据流向，比如在电力系统调度自动化中，其数据自 RTU 单向一发多收，而在综合自动化中则为全双工通信，在故障录波和保护信息组网应用中，则除了全双工通信外，还要满足多方共享信息的需要；在有些情况下，还可能要求某一串口信息的内容同时发往另外几个串口和以太网络。

由于用户的具体应用是多样的，为了更好地满足特定应用的需要，除了选择串口设备以太网络服务器预定的几种工作模式外，我们还可提供定制的服务，具体情况请与我公司技术部门联系。

2、连续的数据传输

数据的连续传输方式，即信息的不间断传输，要求在 DTE、串口设备以太网络服务器和网络中的某一具体应用（计算机/终端上运行的软件）之间保持畅通的数据链路，比如电力系统调度系统、高速公路车流监视系统等。

连续的数据传输还要视信息本身的重要性加以区分，以便选择 TCP/IP 或 UDP/IP 通信协议组，对于重要的信息甚至还需考虑额外的、用户自定义的补充协议，以提高信息传输的可靠性。例如，在电力系统调度自动化系统中，电压（流）的当前值以刷新方式显示即可，也就是说不必关心详尽的变化过程，因此中间丢失几帧信息不会对应用造

成大的影响，但对于开关变位信息是极为重要的，不可丢失的，否则应采用高层软件自行判断加以补充，因此这类信息必须保证信息的完整性。

另外，连续的数据传输若需要满足多方共享的，还必须考虑数据传输的模式和应用之间是否冲突。

3、突发/间断的数据传输

数据的突发/间断传输，可能存在 DTE 方主动、网络中的某一具体应用（计算机/终端上运行的软件）主动、或双方均主动的可能性，要同时满足这些需要，如果全网的 DTE 设备数量在 255 以内，由于计算机/终端机一般拥有更多的资源（如硬件配置、操作系统等），我们建议串口设备以太网络服务器工作在服务器模式，然后由具体应用负责创建和保持链接，如采用多线程编程等；对于超过 255 的，还可以采用分组的办法，把全网的 DTE 设备分成几组，然后采用分时创建、保持和关闭链接（重要 DTE 的链路则可长期保持）的方式来满足链接需求。

采用链接资源（主要指端口和线程）动态变化的“快速扫描”的方法，即计算机/终端机负责数据链路的轮流创建、保持和关闭，可以快速发现 DTE 设备是否有信息需要上送，同时满足信息的主动下发需要，并能更好地节约系统资源，对于用于满足一定数据量传输需要的数据链路则可以由计算机/终端机根据具体情况确定其暂时保持畅通的时间间隔和关闭（端口和线程可以用于与其它的 DTE 创建链路的）时刻。

调试报文通信基本流程

1、通过某种方法进入调试状态/建立调试通信；
2、发送登录报文，并获得授权；
3、完成其他通信；
4、与进入调试状态/建立调试通信相对应的方法结束调试。

说明：只有查看和修改设置参数、设置维护信息、复位 TCP/IP 链接和 CPU 系统时需要预先取得授权，若仅为获得其他信息，如装置工作状态等，可以省去登录授权过程。

/ADJCTL 调试控制

1、将/ADJCTL 置为 0 电平 100ms 后（必须保持 0 电平），模块进入调试状态；
2、按照通信基本流程工作；
3、将/ADJCTL 置为 1 电平 500ms 后（必须保持 1 电平），模块返回工作状态；

网络远程维护

1、与调试端口 61166 或者 61188 建立 TCP 链接；
2、按照通信基本流程工作；
3、关闭与调试端口 61166 或者 61188 建立的 TCP 链接；

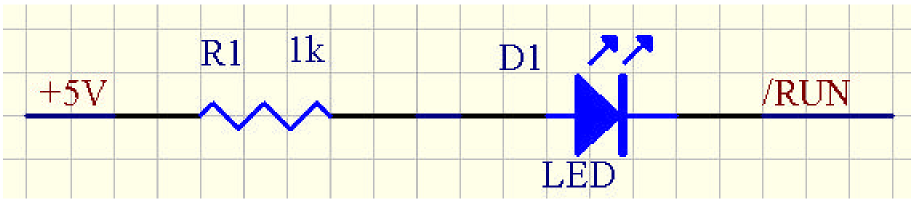
输入/输出

IO1 和 IO2 与 8051 系列单片机的 P1 口相同，是准双向的 I/O，同一引脚在同一时间只能作为输入或者输出，因为输入时无法保持原有输出的状态；若需要保持输出状态，又要读取输入状态时，最好分别采用不同的引脚；若输出和输入可以分时工作，输出无需保持，则输入与输出可以共用同一 I/O 引脚。

IO1 和 IO2 信号的读取与控制是通过维护端口 61166 和 61188 来完成的；具体方法参见后面的通信报文定义。

信号指示

模块有/TPLNK、/NETRX、/NETTX、/RUN 和/ALARM 共 5 个信号指示输出；这些信号可以被单片机所识别，或者用作信号指示 LED 控制；作为信号检测使用时，建议接 10kΩ的上拉电阻；作为 LED 控制时，不用上拉电阻，限流电阻建议采用 1kΩ（**最小必须为 470Ω**），并采用高亮度的 LED；参考下图：



工作模式

N1S01-CORE-5.0 提供 5 种可选的工作模式，具体定义如下：

工作模式 01：仅作为 Server 方式的双向透明传输（远方负责维护数据链路，需要时建立，使用完后可以关闭）；

工作模式 02：仅作为 Client 方式的双向透明传输（本机负责维护数据链路，使数据链路处于常通状态）；需要为串口提供远方用于监听的 IP 地址和端口号；

工作模式 03：仅作为 Client 方式的双向透明传输，串口信息启动链接（本机在串口有信息时主动建立链接，使用完后可以关闭）；需要为串口提供远方用于监听的 IP 地址和端口号；

工作模式 04：作为 Server/Client 方式的双向透明传输（双方共同负责维护数据链路，使数据链路处于常通状态）需要为串口提供远方用于监听的 IP 地址和端口号；

工作模式 05：作为 Server/Client 方式的双向透明传输，串口信息启动链接（双方共同负责维护数据链路，本机在串口有信息时主动建立链接，使用完后可以关闭）；需要为串口提供远方用于监听的 IP 地址和端口号；

其它工作模式：可根据用户的特定需求免费定制；

参数设置

通过串口同步进入设置

运行我公司提供的设置管理软件 N1S Setting，选择好调试串口，在关闭电源的情况下，将 PC 机的串口线和模块的串口（需经过 TTL 到 RS232 的驱动芯片，如 MAX202 等）正确连接接，然后开启电源，N1S Setting 将自动和模块建立通信，这时可以进行参数设置等工作；

本方法主要适合于整机情况下进行参数设置,详细资料请参考 N1S01-232/485 整机使用说明书和 N1S Setting 设置管理软件使用说明书；

通过网络在线设置

在能够正常建立网络通信的情况下（一般属于整机应用或者已经完成二次应用），可以通过 IP 地址和 61166 或 61188 两个调试端口的任何一个构成（IP 、PORT）套接字，建立 TCP 链接，然后可以根据需要使用报文进行通信，如修改参数、模块的工作状态等；

可以采用我公司提供的 N1S NetTools 网管程序进行远程装置管理；用户也可以按照提供的通信报文自行控制，以满足应用和维护软件一体化的要求；

用本方法读取模块工作参数或获取其他信息时不会影响模块的正常工作，如 TCP 链接的建立、关闭和数据通信等，除非修改了模块的工作参数或控制/复位了 CPU 系统；

本方法和整机情况下进行参数设置是相同的，详细资料请参考 N1S01-232/485 整机使用说明书和 N1S Setting 设置管理软件使用说明书；

通过控制/ADJCTL 在线进入设置

将/ADJCTL 置为 0 电平 100ms 后并保持在 0 电平时，模块进入调试状态 ;将/ADJCTL 置为 1 电平 500ms 后并保持 1 电平时，模块返回工作状态；

注意：

- 在 CPU 复位 2 秒内进入调试状态时（通过信息同步），串口参数被自动设为默认的 9600bps、N、8、1，且为全双工方式；从调试状态返回后，串口采用设置中指定的参数和工作模式。
- 在 CPU 复位 2 秒后进入调试状态时（通过/ADJCTL），串口参数不变，即采用正常工作时信息传输的参数进行设置；从调试状态返回后，串口设置不变；从而简化了主 CPU 系统的编程。

几种参数设置方法的比较

	通过串口同步进入设置	通过网络在线设置	通过/ADJCTL 在线进入设置
对模块工作状态的影响	必须进入调试状态；串口参数采用默认的 9600bps、N、8、1；	在不影响正常工作的情况下同时进行调试信息的获取或修改；	必须进入调试状态；串口参数采用正常工作时的参数；
模块可调试状态时间的限制	最长 5 分钟；超过 5 分钟时，自动进入运行状态；	没有时间限制；	最长 5 分钟；超过 5 分钟时，自动进入运行状态；
退出调试状态的条件	发送结束报文或者超时；		/ADJCTL 为 1 状态超过 500ms 或者超时；
完成调试工作后模块的状态	自动进入正常运行状态；	若没有修改工作参数和控制复位 CPU，则模块工作状态不受影响；	自动进入正常运行状态；相当于 CPU 系统复位；
推荐应用情况	我公司提供的整机产品的应用；	适用于任何应用情况；可以远程修改参数、查看或控制模块的工作状态；	主要应用于模块和单片机系统接口，由单片机对模块进行控制；也适用于我公司提供的整机产品的应用；

参数定义

```
typedef struct
{
    // *********************************************
    // * 版本号由程序硬编码，不得通过设置修改 *
```

```
// * ----------------------------------- *
// * 主版本号和副版本号不能同时为 0。        *
// *****************************************
BYTE ucVersion;          // 设置版本号：高 4 位为主版本号，低 4 位为副版本号
BYTE ucDefDateTime[6]; // 设置版本日期：年(-1980)月日时分秒各占一个字节
BYTE ucTypeID;           // 类型定义
BYTE ucReservedVer[2]; // 头信息保留

// ***************************************
// *    以下所有设置均可以通过串口修改     *
// * --------------------------------- *
// * 1.串口设置，2.工作模式，3.网络设置 *
// ***************************************
BOOL blModifyByNetEnabled; // 是否可通过网络改变设置：TRUE-是，FALSE-否
BOOL blModifyByAdjustPin;  // 是否可以通过控制/ADJCTL 进入调试状态 :TRUE-
                           // 是，FALSE-否
BYTE ucReservedCTRL[5];     // 设置控制保留

//////////////
// 串口设置  //
//////////////
BOOL blComEnabled[6];     // 串口使能否：FALSE-否，TRUE-是
BYTE ucComDataBits[6];    // 串口数据位数: 7 或 8
BYTE ucComStopBits[6];    // 串口停止位数: 1 或 2
BYTE ucComParity[6];      // 串口效验方式: 0-无，1-奇 odd，2-偶 even
long lComBaudRate[6];     // 串口波特率
BOOL blComRPEREnable[6]; // 串口奇偶校验错误时，是否替代接收字符；TRUE-
                          // 是，FALSE-否；
BYTE ucComRPERChar[6];   // 串口奇偶校验错误时，接收字符用该字符替代；
                          // 0～255；默认：?
BYTE ucDriverMode[6];    // 外部驱动接口类型：0x00-全双工 RS232，0x80-全
                          // 双工 RS485，0x8F-半双工 RS485
BYTE ucRemoteIP[6][4];   // 对方 IP（仅用于 Client 方式的主动链接），先列
                          // 后行，即[0][0],[0][1],...[0][3],[1][0]...
UINT uiRemotePort[6];     // 对方 PORT（仅用于 Client 方式的主动链接）
BYTE ucRemoteIdleTTL[6]; // 链接状态下通信空闲时间超过该设置时自动关闭连
                        // 接,0-不关闭，其他－秒数
BYTE ucAutoTcpCnnt[6];   // 本串口对应的 socket 是否启用主动申请 TCP 链接
                          // 功能：FALSE-否；TRUE-是；
BYTE ucReservedCom[8];    // 串口保留

/////////////
// 工作模式 //
/////////////
```

```
// 工作模式 01（0x00）：仅作为 Server 方式的双向透明传输
//                   （远方负责维护数据链路，需要时建立，使用完后关闭）
// 工作模式 02（0x01）：仅作为 Client 方式的双向透明传输
//                    （本机负责维护数据链路，使数据链路处于常通状态）
// 工作模式 03（0x02）：仅作为 Client 方式的双向透明传输，
//                    串口信息启动链接（本机在串口有信息时主动建立链接，
//                    使用完后可以关闭）
// 工作模式 04（0x03）：作为 Server/Client 方式的双向透明传输
//                    （双方共同负责维护数据链路，使数据链路处于常通状态）
// 工作模式 05（0x04）：作为 Server/Client 方式的双向透明传输，
//                    串口信息启动链接（双方共同负责维护数据链路，
//                    本机在串口有信息时主动建立链接，使用完后可以关闭）
BYTE ucMode; // 工作模式：0x00-0x0F 厂家保留；0x11-0xFF 为用户定制模式
             // （从 0xFF 开始使用，以使厂家保留不足时可以处理）
BOOL blNeedIoControlGrant; // I/O 输出控制时是否需要登录授权 :TRUE 或 FALSE
BYTE ucReservedMODE[5]; // 模式保留

///////////////////
// 网络接口参数设置 //
///////////////////
BYTE ucReservedNet1[6];    // 网络保留 1
BYTE ucIPv4v6;             // 4-IPv4（默认），6-IPv6
BYTE ucMY_IP_ADDRESS[4];   // 本节点 IP 地址
BYTE ucMY_NETMASK[4];      // 地址解释 MASK
BYTE ucMY_GATEWAY[4];      // 网关 IP 地址
BYTE ucMY_NAMESERVER[4];   // 名字解释服务器 IP 地址
UINT uiMY_MyBasePort1;     // 用于串口基础端口号，ComPort2 = ComPort1 +
                           // 1，...
BYTE ucMaxSockets;         // 最大会话数
UINT uiWaitCloseSec;       // 当 TCP/IP 监听超时或非从容关闭后的自动恢复延
                           // 时（5-9000 秒）
BYTE ucReservedNET[32];    // 网络保留

////////////////////
// TCP/IP 通信控制 //
////////////////////
BYTE ucReLinkTcpCtrl; // 是否允许重新建立/复位已经建立的 TCP 链接：
                      // 0－否（仅对串口链接有效；调试链接不能禁止，
                      // 但其他条件相同）；1－同一 IP 限时；2－同一 IP
                      // 不限时；3－不限制 IP
BYTE ucReservedTCP[9];

////////////////
// 设置维护信息 //
```

```
    ////////////////
    BYTE ucLastDateTime[6];   // 最新修改时间：年(-1980)月日时分秒各占一字节
    BYTE ucDeviceName[13];     // 装置名称：同时作为装置名称和用户访问名称
    BYTE ucDevicePass[9];      // 维护口令
    BYTE ucReservedMain[10];  // 维护信息保留

    // ******************************
    // * CRC 效验码由设置程序生成 *
    // ******************************
    UINT uiCrcValue; // CRC 效验值
} settingType;
```

说明：

1、BYTE 为无符号字节（unsigned char）；UINT 为无符号字（unsigned int）；LONG 为长整形（long）；
2、字、长字、数组的字节顺序与 C 语言相同，低字节在前，高字节在后；
3、IP 地址和掩码的存放顺序为：[0]-第一个字节；如 169.254.1.2，则[0]存放 169，[1]存放 254，[2]存放 1，[3]存放[2]；
4、ucVersion 为程序硬编码的设置版本，写入设置时不得修改原来的内容；否则写入设置报文将应答错误报文；
5、ucTypeID 为程序硬编码的模块类型定义，写入设置时不得修改原来的内容；否则写入设置报文将应答错误报文；
6、ucDriverMode 与 RS485/232、HALF/FULL 和 DE485 引脚信号之间的关系：

<table>
<tr><th>ucDriverMode</th><th>接口模式</th><th>RS485/232</th><th>HALF/FULL</th><th colspan="2">DE485</th></tr>
<tr><td>0x00</td><td>全双工 RS232</td><td>0</td><td>0</td><td colspan="2">1</td></tr>
<tr><td>0x80</td><td>全双工 RS422</td><td>1</td><td>0</td><td colspan="2">1</td></tr>
<tr><td rowspan="2">0x81</td><td rowspan="2">半双工 RS485</td><td rowspan="2">1</td><td rowspan="2">1</td><td>发送</td><td>1</td></tr>
<tr><td>接收</td><td>0</td></tr>
</table>

7、ucRemoteIdleTTL，链接状态下通信空闲时间超过该设置（秒）时自动关闭链接（限于工作模式 03 和 05，即在串口有信息时主动建立的链接）；
8、ucAutoTcpCnnt，本串口对应的 socket 是否启用主动申请 TCP 链接，TRUE-是，FALSE-否（限于工作模式 03 和 05，即在串口有信息时主动建立的链接）；
9、blNeedIoControlGrant，I/O 输出控制时是否需要登录授权，TRUE-是，FALSE-否；授权应用请参考“登录授权”相关部分；
10、uiWaitCloseSec 参数应考虑网络路由情况，若在同一个子网内，可以采用较小的时间设置（如 5 秒），否则应考虑路由带来的延时（如 30 秒）；
11、ucReLinkTcpCtrl 用于限制已经建立 TCP 链接的情况下，是否允许强行重新建立链接，一般地推荐使用 0（不允许），若为了提高链接的可靠性，则可以使用 1（只允许相同的 IP 地址重复链接）；不推荐使用 2（不限制 IP），因为可能导致不同终端（如 PC）之间相互争夺链接，除非能够保证没有争夺情况的发生，或者允许不同终端来强行共享信息（不同终端间由软件/时间来配合完成）；不管是否同一个 IP 地址，若允许重新建立链接，则新的链接申请将导致原有链接的关闭；
12、uiCrcValue 由设置修改方计算，计算范围为除了 uiCrcValue 本身的所有设置

内容；CRC-16 生成公式与通信报文相同，参见相关部分；

13、FALSE 为 0x00，TRUE 为 0xFF（非零）。

通信报文

不管通过任何方式进入参数设置/调试状态，均采用相同的通信报文；具体如下：

```
//////////////////////////////////////////////////////////////////////
//                              定值设定报文定义
// ---------------------------------------------------------------------
//  1．通信格式：9600，8，N，1
//
//  2．报文格式：
//      序号    内容        注释
//        0      EB      --+
//        1      90        | 同步
//        2      EB        | 字符
//        3      90      --+
//        4     STX      内容起始字符
//        5     CMD      报文命令码
//        6     LEN-L    报文正文长度 L
//        7     LEN-H    报文正文长度 H
//             ..正文..
//     HL+8     CRC L    从“报文命令码”开始包括所有正文内容的 CRC 校验码 L
//     HL+9     CRC H    从“报文命令码”开始包括所有正文内容的 CRC 校验码 H
//    HL+10     ETX      内容结束字符
//
//     其中：CRC 校验为 HDLC 标准 16 位 CRC 校验码。它的生成用字节 1 至字节 N 构
//           成信息码的多项式 F（x）除以生成多项式 G（x）所得余数高低位反顺
//           序的结果。F（x）各位排列的次序为：
//                 b0b1b2b3b4b5b6b7b0b1b2b3b4b5b6b7…b0b1b2b3b4b5b6b7
//                     字节 1            字节 2     …      字节 N
//               生成多项式 G（x）=x^16 +x^15 +x^2 +1，F（x）/G（x）所得余
//           数多项式 R（x）：
//                         R（x）=b0x^15 +b1x^14 +…+b14x+b15
//
//  3．特殊报文（下行，上行）
//      下行：指管理 CPU（包括 PC、单片机等）向模块发送信息的方向；
//      上行：指模块向管理 CPU（包括 PC、单片机等）发送信息的方向；
//
//    3.1 ACK 正确应答
//        5     6     报文命令码
//        6     0     报文正文长度 L
//        7     0     报文正文长度 H
//
//    3.2 NAK 否认应答/报文校验错误
```

```
//        5   15H    报文命令码
//        6    0     报文正文长度 L
//        7    0     报文正文长度 H
//
//     3.3 STP 停止通信
//        5    0     报文命令码
//        6    0     报文正文长度 L
//        7    0     报文正文长度 H
//
//     3.4 UKW 错误命令
//        5   FF     报文命令码
//        6    1     报文正文长度 L
//        7    0     报文正文长度 H
//        8   Err    错误代码
//
//   4. 查看装置类型命令报文（下行）
//        5   50H     报文命令码
//        6    0     报文正文长度 L
//        7    0     报文正文长度 H
//
//   5. 上送装置类型命令报文（上行）
//        5   A0H     报文命令码
//        6    6     报文正文长度 L
//        7    0     报文正文长度 H
//        8  Type-L 装置类型代码 L
//        9  Type-H 装置类型代码 H
//       10  PVer-L 程序副版本
//       11  PVer-H 程序主版本
//       12  SVer-L 设置副版本
//       13  SVer-H 设置主版本
//
//   6. 查看设置命令报文（下行）
//        5   51H     报文命令码
//        6    0     报文正文长度 L
//        7    0     报文正文长度 H
//
//   7. 上送设置报文（上行，针对查看设置命令使用）
//        5   A1H     报文命令码
//        6   ??      报文正文长度 L
//        7   ??      报文正文长度 H
//
//   8. 修改设置命令报文（下行）
//        5   52H     报文命令码
//        6   ??      报文正文长度 L
```

```
//          7    ??       报文正文长度 H
//
//   9. 设置修改结果报文（上行，针对修改设置命令使用）
//          5    A2H      报文命令码
//          6    01       报文正文长度 L
//          7    00       报文正文长度 H
//          8    ??       结果标志
//
//  10. 查看日期和时间（下行）
//          5    53H      报文命令码
//          6     0       报文正文长度 L
//          7     0       报文正文长度 H
//
//  11. 上送日期和时间（上行，针对查看日期和时间命令使用）
//          5    A3H      报文命令码
//          6     6       报文正文长度 L
//          7     0       报文正文长度 H
//          8    年       年-1980
//          9    月
//         10    日
//         11    时
//         12    分
//         13    秒
//
//  12. 修改日期和时间（下行）
//          5    54H      报文命令码
//          6     6       报文正文长度 L
//          7     0       报文正文长度 H
//          8    年       年-1980
//          9    月
//         10    日
//         11    时
//         12    分
//         13    秒
//
//  13. 回送日期和时间（上行，针对修改日期和时间命令使用）
//          5    A4H      报文命令码
//          6     6       报文正文长度 L
//          7     0       报文正文长度 H
//          8    年       年-1980
//          9    月
//         10    日
//         11    时
//         12    分
```

```
//         13   秒
//
// 14. 查看程序版本（下行）
//          5   55H      报文命令码
//          6    0       报文正文长度 L
//          7    0       报文正文长度 H
//
// 15. 上送程序版本（上行，针对修改日期和时间命令使用）
//          5   A5H      报文命令码
//          6    6       报文正文长度 L
//          7    0       报文正文长度 H
//          8  PVer-L    程序副版本
//          9  PVer-H    程序主版本
//         10  SVer-L    设置副版本
//         11  SVer-H    设置主版本
//         12  Type-L    装置类型代码 L
//         13  Type-H    装置类型代码 H
//
// 16. 查看程序编译日期（下行）
//          5   56H      报文命令码
//          6    0       报文正文长度 L
//          7    0       报文正文长度 H
//
// 17. 上送程序编译日期（上行，针对修改日期和时间命令使用）
//          5   A6H      报文命令码
//          6    6       报文正文长度 L
//          7    0       报文正文长度 H
//          8   年       年-1980
//          9   月
//         10   日
//         11   时
//         12   分
//         13   秒
//
// 18. 查看程序信息（下行）
//          5   57H      报文命令码
//          6    0       报文正文长度 L
//          7    0       报文正文长度 H
//
// 19. 上送程序信息（上行，针对修改日期和时间命令使用）
//          5   A7H      报文命令码
//          6    ?       报文正文长度 L
//          7    ?       报文正文长度 H
//          8   ...      字符串内容，长度为 HL
```

```
//
// 20. 查看各串口的 TCP 链接状态（下行）
//          5    58H     报文命令码
//          6     0      报文正文长度 L
//          7     0      报文正文长度 H
//
// 21. 上送各串口的 TCP 链接状态（上行）
//          5    A8H     报文命令码
//          6    n+2     报文正文长度 L
//          7     0      报文正文长度 H
//          8    Adj1    Adj1 的 TCP 链接状态
//          9    Adj2    Adj2 的 TCP 链接状态
//         10    COM1    COM1 的 TCP 链接状态
//         11    COM2    COM2 的 TCP 链接状态
//        ...    ...
//      7+n+2    COMn    COMn 的 TCP 链接状态
//
// 22. 查看各串口的 TCP 链接超时值（下行）
//          5    59H     报文命令码
//          6     0      报文正文长度 L
//          7     0      报文正文长度 H
//
// 23. 上送各串口的 TCP 链接超时值（上行）
//          5    A9H       报文命令码
//          6  4*(n+2)     报文正文长度 L
//          7     0        报文正文长度 H
//          8    Adj1-LL   Adj1 的 TCP 链接超时值 LL
//          9    Adj1-L    Adj1 的 TCP 链接超时值 L
//         10    Adj1-H    Adj1 的 TCP 链接超时值 H
//         11    Adj1-HH   Adj1 的 TCP 链接超时值 HH
//         12    Adj2-LL   Adj2 的 TCP 链接超时值 LL
//         13    Adj2-L    Adj2 的 TCP 链接超时值 L
//         14    Adj2-H    Adj2 的 TCP 链接超时值 H
//         15    Adj2-HH   Adj2 的 TCP 链接超时值 HH
//         16    COM1-LL   COM1 的 TCP 链接超时值 LL
//         17    COM1-L    COM1 的 TCP 链接超时值 L
//         18    COM1-H    COM1 的 TCP 链接超时值 H
//         19    COM1-HH   COM1 的 TCP 链接超时值 HH
//        ...    ...
//    7+4*(n+2)    COMn-LL   COMn 的 TCP 链接超时值 LL
// 7+4*(n+2)+1    COMn-L    COMn 的 TCP 链接超时值 L
// 7+4*(n+2)+2    COMn-H    COMn 的 TCP 链接超时值 H
// 7+4*(n+2)+3    COMn-HH   COMn 的 TCP 链接超时值 HH
//
```

```
// 24．复位指定串口的 TCP 链接（下行）
//        5    5AH     报文命令码
//        6    n+1     报文正文长度 L
//        7     0      报文正文长度 H
//        8    res     保留
//        9    ComID1  串口号 1
//       10    ComID2  串口号 2
//      ...    ...
//      8+n    ComIDn  串口号 n
//    ------------------------------------------------------------
//    ComIDx: 0x01-0x7F 为串口号 1-127，0x81-调试链接 1，0x82-调试链接 2,
//            0x00-全部串口，0xF0-全部调试链接，0xFF-全部链接,
//
// 25．将要复位指定串口的 TCP 链接（上行）
//        5    AAH     报文命令码
//        6    n+1     报文正文长度 L
//        7     0      报文正文长度 H
//        8     FLAG   结果标志
//        9    ComID1  串口号 1
//       10    ComID2  串口号 2
//      ...    ...
//      8+n    ComIDn  串口号 n
//    ------------------------------------------------------------
//    FLAG:  0x00-成功（其后内容有效），
//           0xE1-复位链接的串口数错误（其后内容无效），
//           0xE2-复位链接的串口编号错误（其后内容无效）
//    ------------------------------------------------------------
//    ComIDx: 0x01-0x7F 为串口号 1-127，0x81-调试链接 1，0x82-调试链接 2,
//            0x00-全部串口，0xF0-全部调试链接，0xFF-全部链接,
//
// 26．查看设置维护信息（下行）
//        5    5BH     报文命令码
//        6     0      报文正文长度 L
//        7     0      报文正文长度 H
//
// 27．上送设置维护信息（上行）
//        5    ABH     报文命令码
//        6    23H     报文正文长度 L
//        7     0      报文正文长度 H
//        8    ID      装置 ID 号
//        9     0      分隔符 0
//       10     0      分隔符 0
//       11    yy      最新修改时间：年-1980
//       12    mm      最新修改时间：月
```

```
//      13   dd      最新修改时间：日
//      14   hh      最新修改时间：时
//      15   mm      最新修改时间：分
//      16   ss      最新修改时间：秒
//      17    0      分隔符 0
//      18    0      分隔符 0
//      19   nm0     装置名称字节[0]
//     ...   ...     ...
//      30   nm11    装置名称字节[11]
//      31    0      分隔符 0
//      32    0      分隔符 0
//      33   ps0     维护口令字节[0]
//     ...   ...
//      40   ps15    维护口令字节[7]
//      41    0      结束符 0
//      42    0      结束符 0
//
// 28. 查看装置工作状态（下行）
//       5   5CH     报文命令码
//       6    0      报文正文长度 L
//       7    0      报文正文长度 H
//
// 29. 上送装置工作状态（上行）
//       5   ACH     报文命令码
//       6   n+4     报文正文长度 L
//       7    0      报文正文长度 H
//       8   FLAG1   异常状态 1（参考：装置异常工作状态定义）
//       9   FLAG2   异常状态 2（参考：装置异常工作状态定义）
//      10   Adj1    Adj1 的 TCP 链接状态
//      11   Adj2    Adj2 的 TCP 链接状态
//      12   COM1    COM1 的 TCP 链接状态
//      13   COM2    COM2 的 TCP 链接状态
//      ...  ...
//   7+n+4   COMn    COMn 的 TCP 链接状态
//
// 30. 授权登录申请（下行）
//       5   5DH     报文命令码
//       6   OBH     报文正文长度 L
//       7    0      报文正文长度 H
//       8   ps0     维护口令字节[0]
//     ...   ...
//      15   ps7     维护口令字节[7]
//      16    0      分隔符 0
//      17    0      分隔符 0
```

```
//        18  TTL       存活周期（15～255 秒）
//
// 31．授权登录回复（上行）
//         5   ADH      报文命令码
//         6   10H      报文正文长度 L
//         7    0       报文正文长度 H
//         8  nm0       装置名称字节[0]
//       ...  ...       ...
//        19  nm15      装置名称字节[11]
//        20    0       分隔符 0
//        21    0       分隔符 0
//        22  TTL       实际批准的 TTL
//        23  FLAG      批准结果：0-已批准；0x80-存活周期无效（将自动设为
//                                上/下限）；0xF1-装置口令错误
//
// 32．复位 CPU 系统（下行）
//         5   5FH      报文命令码
//         6    0       报文正文长度 L
//         7    0       报文正文长度 H
//
// 33．将要复位 CPU 系统（上行）
//         5   AFH      报文命令码
//         6    0       报文正文长度 L
//         7    0       报文正文长度 H
//
// 34．读取 I/O 状态输入（下行）
//         5   60H      报文命令码
//         6    1       报文正文长度 L
//         7    0       报文正文长度 H
//         8   IOn      IO 编号：IO1-1，IO2-2，...
//
// 35．返回 I/O 状态输入当前值（上行）
//         5   B0H      报文命令码
//         6    2       报文正文长度 L
//         7    0       报文正文长度 H
//         8   IOn      IO 编号：IO1-1，IO2-2，...
//         9   VALUE    当前状态值：0x00－“0”电平，0xFF－“1”电平，
//                                  0x8F-错误的 IO 编号
//
// 36．用指定内容写入 I/O（控制输出，下行）
//         5   61H      报文命令码
//         6    2       报文正文长度 L
//         7    0       报文正文长度 H
//         8   IOn      IO 编号：IO1-1，IO2-2，...
```

```
//        9   VALUE  输出状态值：0x00－“0”电平，0xFF－“1”电平
//
// 37. 返回写入指定 I/O 的内容（仅作为应答之用，上行）
//        5   B1H    报文命令码
//        6    2     报文正文长度 L
//        7    0     报文正文长度 H
//        8   IOn    IO 编号：IO1-1，IO2-2，...
//        9   VALUE  输出状态值：0x00－“0”电平，0xFF－“1”电平，
//                               0x8F-错误的 IO 编号
//////////////////////////////////////////////////////////////////////
#define cnCmdOffset          5 // 命令码在报文中的偏移
#define cnLenLOffset         6 // LEN-L 在报文中的偏移
#define cnLenHOffset         7 // LEN-H 在报文中的偏移
#define cnCrcBeginOffset     5 // CRC 计算的首字符在报文中的偏移
#define cnContentOffset      8 // 报文内容起始偏移
#define cnCrcLOffset         8 // CRC-L 在报文中的偏移
#define cnCrcHOffset         9 // CRC-H 在报文中的偏移
#define cnCrcPreChars        3 // CRC 计算时除报文长度指出的额外字符数：
                              // 命令码（1）+ 正文长度（2）
#define cnEtxOffset         10 // ETX 在报文中的偏移
#define cnMinLenOfFrame     10 // 发送（接收）实际可能的最小报文长度
#define cnMaxLenOfFrame     cnMaxSetting + 48 // 发送（接收）实际可用的
                                              // 最大报文长度

// 特殊报文
#define cnSpeStpSet      0x00 // STP 停止通信（下行，上行）
#define cnSpeAckSet      0x06 // ACK 正确应答（下行，上行）
#define cnSpeNakSet      0x15 // NAK 否认应答（下行，上行）
#define cnSpeUkwSet      0xFF // UKW 错误命令（下行，上行）

// 命令报文
#define cnCmdReadTypeID     0x50 // 查看装置类型命令报文（下行）
#define cnCmdReadSet        0x51 // 查看设置命令报文（下行）
#define cnCmdWriteSet       0x52 // 修改设置命令报文（下行）
#define cnCmdReadTime       0x53 // 查看日期和时间（下行）
#define cnCmdWriteTime      0x54 // 修改日期和时间（下行）
#define cnCmdReadPrgVer     0x55 // 查看程序版本（下行）
#define cnCmdReadPrgTime    0x56 // 查看程序编译日期（下行）
#define cnCmdReadPrgInfor   0x57 // 查看程序信息（下行）
#define cnCmdReadTcpStatus  0x58 // 查看各串口的 TCP 链接状态（下行）
#define cnCmdReadTcpTimeCN  0x59 // 查看各串口的 TCP 链接超时值（下行）
#define cnCmdResetTcpByCOM  0x5A // 复位指定串口的 TCP 链接（下行）
#define cnCmdRdSetMainInfo  0x5B // 查看设置维护信息（下行）
#define cnCmdReadStatus     0x5C // 查看装置工作状态（下行）
```

```
#define cnCmdLoginRequest   0x5D // 授权登录申请（下行）
#define cnCmdResetCpuHard   0x5F // 复位 CPU 系统（下行）
#define cnCmdReadIOValue    0x60 // 读取 I/O 状态输入（下行）
#define cnCmdWriteIOValue   0x61 // 用指定内容写入 I/O（下行）

// 应答报文
#define cnAnsReadTypeID     0xA0 // 上送装置类型命令报文（上行）
#define cnAnsReadSet       0xA1 // 上送设置报文（上行，针对查看设置命令使用）
#define cnAnsResultSet     0xA2 // 设置修改结果报文（上行，针对修改设置命令）
#define cnAnsReadTime       0xA3 // 上送日期/时间（上行，针对查看时间命令）
#define cnAnsWriteTime      0xA4 // 回送日期/时间（上行，针对修改时间命令）
#define cnAnsPrgVer        0xA5 // 上送程序版本（上行，针对修改日期/时间命令）
#define cnAnsPrgTime       0xA6 // 上送程序编译日期（上行，针对查看程序编译
                                // 日期命令）
#define cnAnsPrgInfor      0xA7 // 上送程序信息（上行，针对查看程序信息命令）
#define cnAnsTcpStatus      0xA8 // 上送各串口的 TCP 链接状态（上行）
#define cnAnsTcpTimeCN      0xA9 // 上送各串口的 TCP 链接超时值（上行）
#define cnAnsResetTcpByCOM  0xAA // 将要复位指定串口的 TCP 链接（上行）
#define cnAnsSetMainInfo    0xAB // 上送设置维护信息（上行）
#define cnAnsReadStatus     0xAC // 上送装置工作状态（上行）
#define cnAnsLoginRequest   0xAD // 授权登录回复（上行）
#define cnAnsResetCpuHard   0xAF // 将复位 CPU 系统（上行）
#define cnAnsReadIOValue    0xB0 // 读取 I/O 状态输入（上行）
#define cnAnsWriteIOValue   0xB1 // 用指定内容写入 I/O（上行）

// 复位 TCP 结果常量定义
#define cnRstTcpComCntOk      0x00 // 成功（其后内容有效）
#define cnRstTcpComCntErr     0xE1 // 复位链接的串口数错误（其后内容无效）
#define cnRstTcpComIdErr     0xE2 // 复位链接的串口编号错误（其后内容无效）

// 登录授权常量定义
#define cnLoginOk            0x00 // 批准结果：0x00-已批准
#define cnLoginTtlFloorErr  0x80 // 批准结果：0x80-已批准，但存活周期无效（小
                                // 于 15 秒时将自动设为下限 15 秒）
#define cnLoginPassErr       0xF1 // 批准结果：0xF1-装置口令错误

//  TCP 状态定义
#define cnTcpStateListen         0 // 监听（没有链接的空闲状态）
#define cnTcpStateSynRcvd        1 // 接收到同步信号（TCP 链接申请）
#define cnTcpStateSynSent        2 // 发送同步信号（响应 TCP 链接申请）
#define cnTcpStateEstablished    3 // 链接已经建立
#define cnTcpStateFinWait1       4 // 从容关闭等待状态 1
#define cnTcpStateFinWait2       5 // 从容关闭等待状态 2
#define cnTcpStateClosing        6 // 正在关闭
```

```
#define cnTcpStateCloseWait      7 // 等待关闭
#define cnTcpStateLastAck        8 // 关闭前的最后一个确认报文
#define cnTcpStateClosed         9 // 已经关闭
#define cnTcpStateTimeWait      10 // 执行关闭后的等待对方确认关闭报文
#define cnTcpStateNeverUsed    255 // 自 CPU 复位以来还未链接过

//////////////////
// 装置状态监测 //
//////////////////
// 装置异常状态字节 1 和 装置异常状态字节 2

// 装置异常工作状态定义（对应位为“1”表示存在该异常信息）
#define cnFailStaExtCnctF10  0x01 // FLAG1.0，外部网络链接失败
#define cnFailStaEepromF11   0x02 // FLAG1.1，EEPROM 异常
#define cnFailStaEepromRdF12 0x04 // FLAG1.2，EEPROM 读异常
#define cnFailStaEepromWrF13 0x08 // FLAG1.3，EEPROM 写异常
#define cnFailStaSettingF14  0x10 // FLAG1.4，EEPROM 中的设置异常（CRC、装置
                                  // 类型或设置版本错误）
#define cnFailSta___F15      0x20 // FLAG1.5，
#define cnFailSta___F16      0x40 // FLAG1.6，
#define cnFailSta___F17      0x80 // FLAG1.7，
#define cnFailStaEtherIcF20  0x01 // FLAG2.0，网络芯片异常（重发缓冲溢出）
#define cnFailSta___F21      0x02 // FLAG2.1，
#define cnFailSta___F22      0x04 // FLAG2.2，
#define cnFailSta___F23      0x08 // FLAG2.3，
#define cnFailSta___F24      0x10 // FLAG2.4，
#define cnFailSta___F25      0x20 // FLAG2.5，
#define cnFailSta___F26      0x40 // FLAG2.6，
#define cnFailSta___F27      0x80 // FLAG2.7，

// 报文错误代码定义
#define cnWrongCmdErr        0xE0 // 错误的命令码
#define cnTcpSetDisabledErr  0xE1 // 不能通过网络 TCP 修改设置
#define cnTcpNotGrantErr     0xE2 // 未登录授权
#define cnContentLenErr      0xE3 // 报文内容长度错误

// 设置修改结果代码定义
#define cnSettingWriteOK    0x06 // 设置修改成功
#define cnSettingLenError   0x80 // 设置长度错误
#define cnSettingCrcError   0x81 // 设置 CRC 错误
#define cnSettingVerError   0x82 // 设置版本错误
#define cnSettingTypeIDErr  0x83 // 装置类型错误
#define cnReadEepromError   0xF0 // 读 EEPROM 错误
#define cnWriteEepromError  0xF1 // 写 EEPROM 错误
```

```
#define cnRdWrEepromError   0xFF // 读写 EEPROM 错误
```

CRC 计算 C 程序

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

// 计算数据块的 16 位 CRC 值
unsigned int IntCRC16(unsigned char data[], unsigned int LenOfData)
{
  char CRC_Table[64] =
  {0x00, 0x00, 0xC0, 0xC1, 0xC1, 0x81, 0x01, 0x40, 0xC3, 0x01, 0x03,
   0xC0, 0x02, 0x80, 0xC2, 0x41, 0xC6, 0x01, 0x06, 0xC0, 0x07, 0x80,
   0xC7, 0x41, 0x05, 0x00, 0xC5, 0xC1, 0xC4, 0x81, 0x04, 0x40, 0x00,
   0x00, 0xCC, 0x01, 0xD8, 0x01, 0x14, 0x00, 0xF0, 0x01, 0x3C, 0x00,
   0x28, 0x00, 0xE4, 0x01, 0xA0, 0x01, 0x6C, 0x00, 0x78, 0x00, 0xB4,
   0x01, 0x50, 0x00, 0x9C, 0x01, 0x88, 0x01, 0x44, 0x00};

  int a = 0, b = 0, r2 = 0, r3 = 0, r4 = 0, r5 = 0;
  int r6 = 0, i = 0, med1 = 0, med2 = 0;

  for (i = 0; i < LenOfData; i ++)
  {
      a = data[i];
      a = (a ^ r5) & 0xff;
      b = a;
      a = a * 2;
      a = (a & 0x1e) & 0xff;
      r4 = a;
      a = a + 1;
      a = CRC_Table[a];
      r2 = a;
      a = r4;
      a = CRC_Table[a];
      r3 = a;
      a = b;

      med1 = (a & 0x0f) * 16;
      med2 = (a & 0xf0) / 16;
      a = med1 + med2;

      a = a * 2;
      a = (a & 0x1e) & 0xff;
      a = a + 32;
```

```
        r4 = a;
        a = CRC_Table[a];
        b = a;
        a = r4;
        a = a + 1;
        a = CRC_Table[a];
        a = (a ^ r2) & 0xff;
        a = (a ^ r6) & 0xff;
        r5 = a;
        a = b;
        a = (a ^ r3) & 0xff;
        r6 = a;
    }

    return r6 * 256 + r5;
}

void main()
{
  unsigned char ucBuffer[] =
  // 0    1     2    3    4    5    6    7    8   9    10
  {0x68, 0x01, 0x01, 0x01, 0x02, 0x06, 0x63, 0x3A, 0x5C, 0x2A, 0x2E,
  // 11   12    13
   0x2A, 0x8D, 0x13};

  clrscr();
  int intCrcValue = IntCRC16(ucBuffer + 1, 11);
  printf("\n CRC = (%6d, 0x%04X)", intCrcValue, intCrcValue);
  getch();
}
```

登录授权

“登录授权”就是通过调试报文把装置口令与 N1S01 事先（以前）设置的口令进行对照，若相同，则允许执行所有报文要求的工作，否则 N1S01 拒绝执行部分重要功能；

当进行以下工作时，需要事先进行登录，以获得执行重要信息的读取、修改或 I/O 控制输出（受 blNeedIoControlGrant 参数的控制）的权利，否则 N1S01-CORE 将拒绝执行（返回错误代码为 cnTcpNotGrantErr 未登录授权的 cnSpeUkwSet 错误报文）：

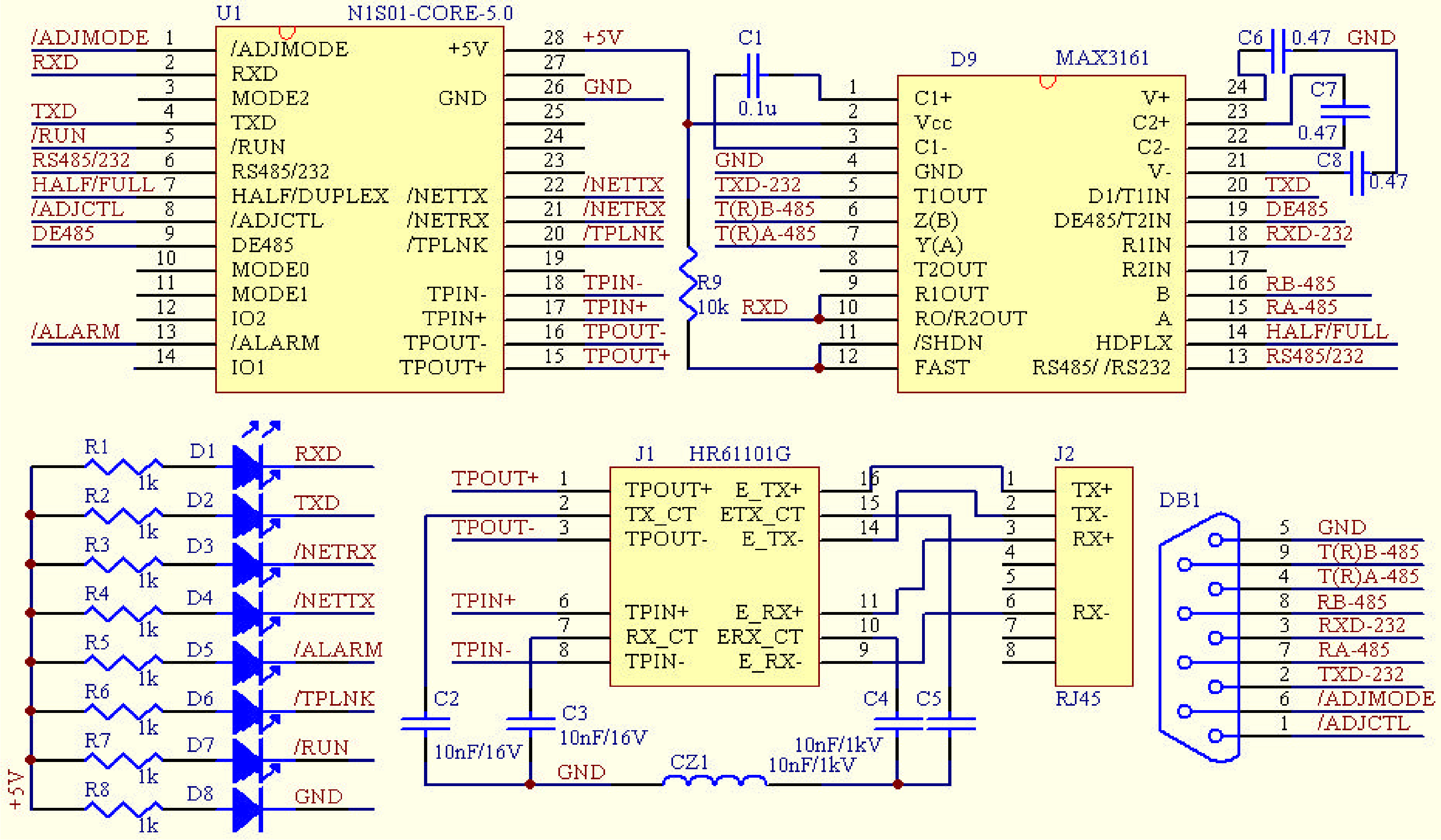
- cnCmdReadSet，查看设置命令报文
- cnCmdWriteSet，修改设置命令报文
- cnCmdResetTcpByCOM，复位指定串口的 TCP 链接
- cnCmdRdSetMainInfo，查看设置维护信息
- cnCmdResetCpuHard，复位 CPU 系统
- cnCmdWriteIOValue，I/O 输出（受 blNeedIoControlGrant 参数的控制）

获得登录授权的方法：

1、通过某种方法使 N1S01-CORE 进入调试状态，或通过网络建立调试链接；
2、向 N1S01-CORE 发送 cnCmdLoginRequest 授权登录申请报文；
3、N1S01-CORE 返回 cnAnsLoginRequest 授权登录回复报文,从而得知是否登录成功；若成功，则执行步骤 4，否则应检查口令是否正确；
4、可以在登录授权的时间内执行任何命令；

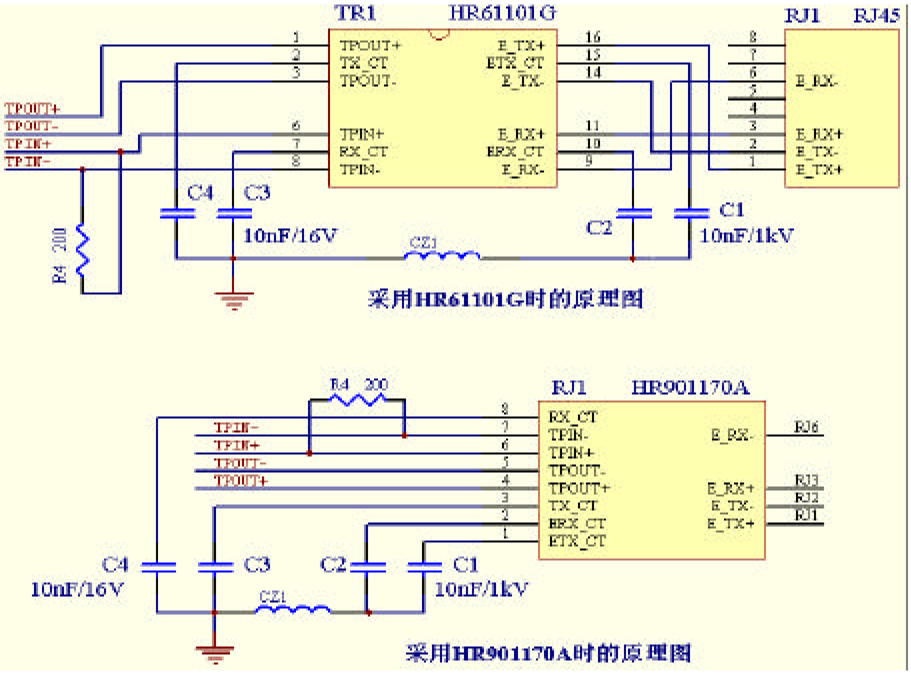
注意：登录报文可以重复发送，N1S01-CORE 从最后一次授权成功开始倒计时，总时间为最后一次授权成功所赋予“存活周期（有效期）”; 当倒计时至 0 时，需要重新申请授权。

应用实例 01－单口串口服务器整机 N1S01-232/485 的设计和应用
（原理图）

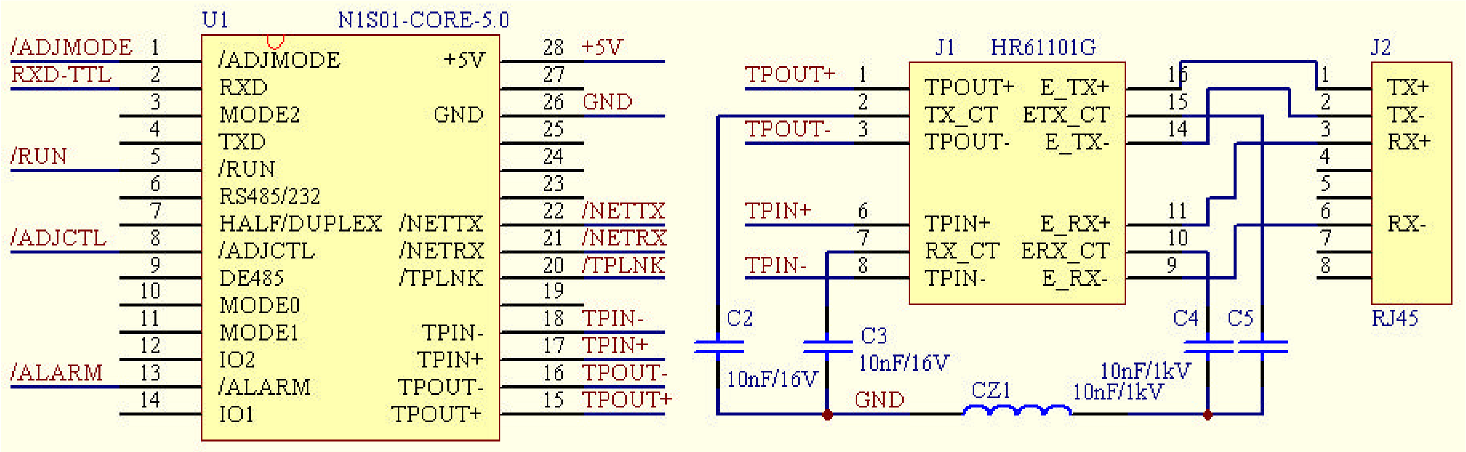


应用实例 01－单口串口服务器整机 N1S01-232/485 的设计和应用（原理说明）

1、 这里我们采用 N1S01-CORE-5.0 模块后，只需要一片外围驱动芯片 MAX3161 就构成了单口串口服务器的整机，而且可以同时满足全双工 RS232、半双工 RS485 和全双工 RS422（RS485 电气兼容）的选择应用要求；其中 CZxx 是磁珠，用于抑制干扰；
2、 选择全双工 RS232 时，使用 DB15 上的 RXD-232、TXD-232 和 GND 三个信号；选择全双工 RS422/485 时，使用 T(R)A-485、T(R)B-485、RA-485 和 RB-485；选择半双工 RS485 时，使用 T(R)A-485、T(R)B-485；
3、 HR61101G 为 10BaseT 网络滤波器，我公司可以配套提供；当然也可以使用其他兼容的网络滤波器；另外，还可以选配我公司提供的 HR901170A，HR901170A 是内部封装了网络滤波器的 RJ45 插座（外部为金属封装），这样可以进一步减少整机的体积；
4、 可以选择其他的接插件替代 DB9 和 RJ45 插座，以满足不同结构设计的要求；
5、 我们在这里提供了电源指示、装置运行指示（/RUN）、网络外部接线状态指示（/TPLNK）、装置异常指示（/ALARM）、网络数据包接收指示（/NETRX）、网络数据包发送指示（/NETTX）、串行口发送指示（TXD）和串行口接收指示（RXD）；其中，由于 UART 处于停止发送状态时 TXD 电平为 1，发送起始位时为 0，另外数据中可能含有 0 位，因此 UART 发送时 TXD 将闪烁；RXD 的闪烁是因为串口通信的对方发送了数据；
6、 实际应用中，可能需要在网络和串口之间实现电气隔离，以提高整机的抗干扰能力和确保网络系统的安全；这时只需要把 TTL 信号经过光耦器件即可，其中 TXD 和 RXD 推荐采用 6N137，其他信号可以采用价格低廉的 TLP521 系列器件；另外，与户外设备链接时，还应在引入线之间加上瞬变抑制二极管，如 P6KExxCA 系列（双向保护）， RS232 信号采用 P6KE15CA， RS485 和其他的 5V 信号采用 P6KE6CA，以提高整机系统的突波保护能力；瞬变抑制二极管应装在尽量靠近信号输入接插件的位置；
7、 该应用无需编写任何程序代码，可以方便地满足现有智能设备的网络接入改造要求；
8、 模块参数设置方法可以采用前面相关部分所提到的任何一种；
9、 采用 HR61101G 和 HR901170A 时，网络接口电路如下图：

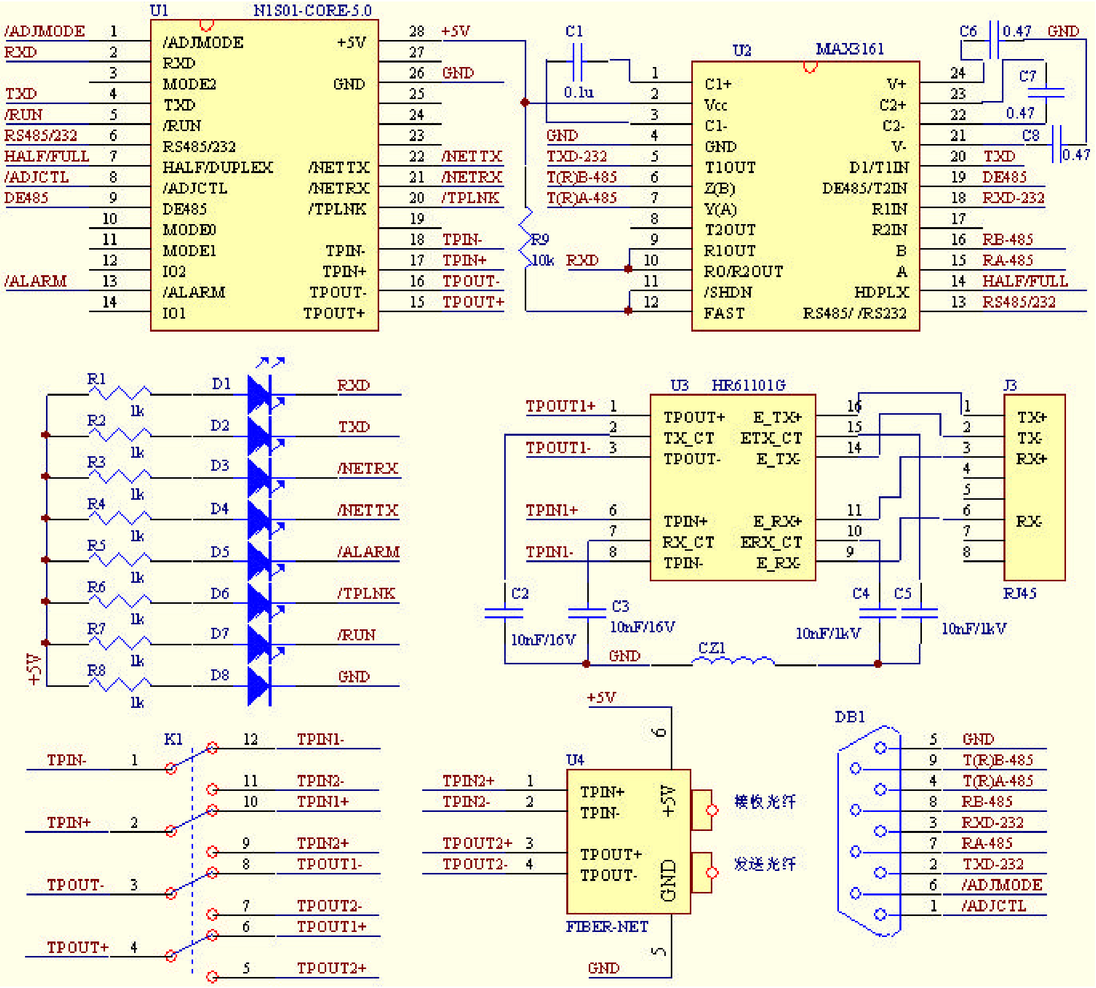


应用实例 02－N1S01-CORE-5.0 在 GPS 网络接口中的应用



1、考虑到 GPS 通过串口提供时间信息是单向的（仅输出），接入 GPS 装置或接收模块的 TXD 信号即可；因为同在装置内部，可以直接采用 TTL 接口；若需要考虑 GPS 装置/外部天线和网络之间的电气隔离，则应在网络模块与 GPS 装置或接收模块的信号之间加光耦器件，建议采用 6N137，若波特率低于 9600bps，则可以采样价格低廉的 TLP521 系列器件；
2、因为时间信息是单向传输的，因此可以考虑串口信息的共享，如多个客户机/终端通过 TCP/IP 共享串口的时间信息；可以通过选择 N1S01-CORE-5.0 的工作模式二实现串口信息的一发多收；
3、一般地，除了提供网络接口外，GPS 装置还要同时满足 RS232、RS485/422 接口；另外，输出的串口信息也是经过 MCU 翻译/解释的；实际应用中，N1S01-CORE-5.0 的 RXD-TTL 信号一般并联接在 MCU 的 UART 输出信号上，和 RS232、RS485/422 共享 TXD（TTL）；因此，可以通过 GPS 装置的主 MCU 来完成 N1S01-CORE-5.0 的参数设置和状态控制，设置方法参见前面有关部分的说明；当然，也可以采用前面描述的其他设置方法；
4、状态信号/ALARM、/NETTX、/NETRX、/TPLNK 可以被 GPS 装置的主 MCU 识别，也可以直接驱动 LED 显示状态；
5、如果 GPS 装置的机箱较大，或者说，网络信号的输出线较长，建议采用上图中网络变压器和 RJ45 分离的方式，以提高网络通信的可靠性；若 N1S01-CORE-5.0 与 RJ45 的距离很近，也可以采用网络变压器和 RJ45 一体化的 HR901170A；具体应用原理图参见应用实例 01（原理说明）；
6、若不需要额外的 RS232、485/422 输出信号，N1S01-CORE-5.0 可以完成 GPS 接收模块的信息翻译/解释，以节省 GPS 装置的成本；若需要同时具备 RS232、485/422 输出，则可以选择我公司生产的 N1S02-CORE-3.0 来完成该功能，一个串口用于接收 GPS 接收模块的信息，另一个串口用于输出解释/翻译后的时间信息；若需要此功能，请与我公司技术部门联系定制服务事宜；

应用实例 03－N1S01-CORE-5.0 在电力系统配电自动化中的应用

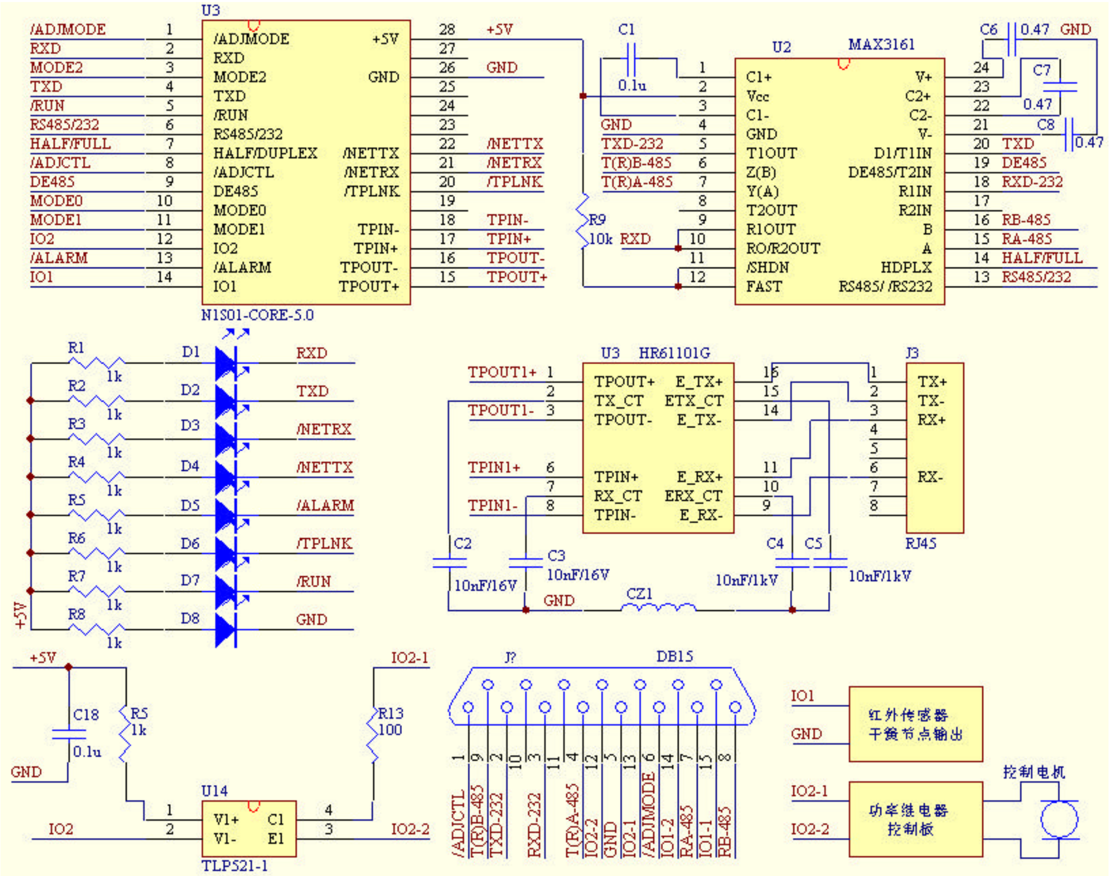


1、 电力系统配电自动化的通信需要适应现场不同情况，如采用电力低压载波、RS485/422、光 MODEM 和光纤以太环网等，因此需要一种灵活的通信接入解决方案；

2、 上图中，同时考虑了常规的几种接入方案，以供实际应用选择；其中，RJ45 电气网络接口和光纤网络接口的信号是通过 4 位单刀双掷拨码进行切换的；光纤网络接口模块由我公司提供，可以实现多模 2km、单模 5km、单模 20km、单模 60km 等常用接入，还可以选择单模的单芯收发模块，以节约光缆；

3、 其他实现原理请参考前面的应用实例说明。

应用实例 04－N1S01-CORE-5.0 在自动门禁系统中的应用



1、 本应用中，使用了 IO1 和 IO2 作为远程检测和控制；其中 IO1 用于检测红外传感器或其他机械机构提供的状态输出，以确定当前自动门（窗）的位置（如开、关）；IO2 用于控制电机打开/关闭门（窗）；
2、 RS232、RS485/422 可以用于传输数字摄像头提供的监视信息，以提供外景；
3、 可以将网络接口与就近的局域网插座链接起来，以简化安装布线、扩展和维护工作；
4、 监控系统的软件可以安装在局域网的任何 PC 机上，也可实现多方共享信息，并通过授权（如密码登录等方法）允许某些操作员具有控制权限；
5、 其他实现原理请参考前面的应用实例说明。